Quantum Information and Computation, Vol. 0, No. 0 (2003) 000–000
© Rinton Press

BLIND RECONCILIATION

JESUS MARTINEZ-MATEO, DAVID ELKOUSS, VICENTE MARTIN[a]
Research group on Quantum Information and Computation
Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain

Received (received date)
Revised (revised date)

Information reconciliation is a crucial procedure in the classical post-processing of quantum key distribution (QKD). Poor reconciliation efficiency, revealing more information than strictly needed, may compromise the maximum attainable distance, while poor performance of the algorithm limits the practical throughput in a QKD device. Historically, reconciliation has been mainly done using close to minimal information disclosure but heavily interactive procedures, like *Cascade*, or using less efficient but also less interactive —just one message is exchanged— procedures, like the ones based in low-density parity-check (LDPC) codes. The price to pay in the LDPC case is that good efficiency is only attained for very long codes and in a very narrow range centered around the quantum bit error rate (QBER) that the code was designed to reconcile, thus forcing to have several codes if a broad range of QBER needs to be catered for. Real world implementations of these methods are thus very demanding, either on computational or communication resources or both, to the extent that the last generation of GHz clocked QKD systems are finding a bottleneck in the classical part. In order to produce compact, high performance and reliable QKD systems it would be highly desirable to remove these problems. Here we analyse the use of short-length LDPC codes in the information reconciliation context using a low interactivity, *blind*, protocol that avoids an a priori error rate estimation. We demonstrate that 2×10^3 bits length LDPC codes are suitable for blind reconciliation. Such codes are of high interest in practice, since they can be used for hardware implementations with very high throughput.

Keywords: Quantum key distribution, information reconciliation, low-density parity-check codes, rate-compatible, interactive reconciliation, short-length codes
Communicated by: to be filled by the Editorial

1 Introduction

Quantum key distribution (QKD) [1] is a process that runs in two phases: a quantum and a classical one. The second one distills a secret key from the quantum signals produced and transmitted in the first. The distillation process begins with an information reconciliation step: from two correlated strings at both ends of the channel, a common string is extracted by publishing some amount of information.

In this context, a set of outputs obtained from measurements of quantum states, must be cooperatively processed at both ends of the transmission channel in order to obtain the common secret key from which all information leakage has been bounded [2]. The set of

[a] vicente@fi.upm.es

measurements —raw key— have errors due to imperfections in the devices, decoherence, eavesdropping or the limited efficiency of the protocol itself. These data strings must be reconciled in order to have the same string at both ends, a process that by revealing information through a public but noiseless channel removes any discrepancy and is known as *information reconciliation* [4].

In order to further clean the reconciled key from any information leakage, either by the previous procedure or by the exchange of quantum signals, a second step is required. This is known as *privacy amplification* [5] and generates the final secret key, shorter but with a known upper bound on the information leakage. Thus, the final secret key length is dependent also on the quality of the information reconciliation step and its behavior regarding security related parameters (e.g. finite size effects).

Information reconciliation in QKD is a problem already addressed by the authors of the original BB84 protocol [3, 4, 6]. In the pioneering BBBSS protocol [3], the authors propose a reconciliation protocol based in the exchange of a number of parities (syndromes). If any parity-check equation of an exchanged syndrome is not verified, the parties carry out a dichotomic search to find the corresponding error. Note that in each parity-check equation an odd number of errors can be detected, but only one of them can be corrected using a binary search. The procedure works iteratively, shuffling the bits of the key to reconcile and exchanging successive syndromes. Later, in Ref. [4], the authors realized that each located error produces side information that can be used with a previously exchanged syndrome. The new protocol was called *Cascade* in reference to the iterative or cascading process of identifying errors. Several optimizations were proposed for the BBBSS and *Cascade* protocols [7–9]. However, all these protocols are highly interactive since they require many communication rounds: the parties have to exchange a large number of messages. Despite its interactivity, *Cascade* continues being one of the most widely used protocols for information reconciliation in QKD, probably due to its simplicity and relatively good efficiency.

Other protocols have been proposed in the literature. For instance, in *Winnow* the authors use Hamming codes for the calculation of separate syndromes instead of a simple parity-check equation [10]. However, the efficiency of this protocol is still far from the Shannon limit.

As early as 2003, the Alamos group hinted at the use of parity-checks as in telecommunication systems [11], but the group did not present any result referring to the use of low-density parity-check (LDPC) codes until one year later [12, 13]. This is one of the first applications of the modern coding theory to the information reconciliation problem in QKD.

The objective of this work is to produce an information reconciliation method amenable to practical implementation using modern hardware for embedded systems. We base our approach in the use of rate adaptive low-density parity-check (LDPC) codes. In Ref. [14] we propose a reconciliation protocol for large codes (2×10^5) in order to obtain the best possible efficiency. These codes have the advantage of minimal interactivity, thus avoiding one of the main disadvantages of *Cascade*. In Ref. [15] we studied the efficiency improvement when relaxing the condition of minimal interactivity, also for large codes. However, neither method is appropriate for a hardware implementation. The purpose of the present paper is to adapt those methods in order to cover a varying error range with high efficiency and, at the same time, make them suitable for a hardware implementation. The limited resources available in embedded hardware make unfeasible the use of a long-length code, thus we focus our interest in the use of short-length LDPC codes. Here we will be using a code length as short as 2×10^3, two orders of magnitude smaller than in the previous works. This implies that the techniques used in the previous algorithms are no longer optimal. For instance, the random puncturing methods used previously can reduce to zero the distillable secret key rate, thus new schemes had to be devised [26]. Moreover, we demonstrate in this correspondence that by allowing a limited interactivity, the final efficiency of the reconciliation can improve significantly. In Ref. [15] we studied the limiting case where only one bit changes per step, without regard of minimizing the number of steps. Here we minimize the interactivity, demonstrating that as little as three messages can approach the efficiency of the maximally interactive case. This makes possible a relatively high throughput process implementable with limited resources. For example, three relatively small FPGA blocks that can work in parallel will suffice. Remarkably, the new protocol works without an a priori estimation of the quantum bit error rate, a fact with interesting implications in finite key analysis: since no extra information is revealed, it does not need to be subtracted from the key.

This correspondence is organised as follows. In Section 2 we introduce the information reconciliation problem and its application using low-density parity-check codes. In Section 3 we review some techniques used for adapting the information rate of a correcting code, and reduce the information disclosed in the reconciliation when using LDPC codes. In Section 4 we present an interactive version of a rate-adaptive protocol (that we call *blind*) that improves the average efficiency. In Section 5 we show some simulation results of the blind protocol using short-length LDPC codes. Finally, in Section 6 we present our conclusions.

2 Information Reconciliation and Channel Coding

In this section we consider the problem of information reconciliation from an information theoretic perspective and study some figures of merit relevant to the discussed protocols.

Information reconciliation is the generic name of any method used to ensure that two parties agree on a common string provided they have access to two correlated sequences X and Y [16]. During reconciliation the two parties exchange a set of messages M over a noiseless channel such that at the end of the process they agree on some string function of their strings and the exchanged messages. In our case, the correlated strings are obtained by Alice and Bob after the quantum phase of the QKD protocol has finished. It does not matter whether an actual quantum channel has been used to transmit qubits from Alice to Bob as in a standard prepare and measure protocol or an entangled pairs emitter acts as the source of correlations. In both cases, X and Y can be regarded as correlated random variables and every symbol in Y can be seen as given by transition probability $p_W(y|x)$, or equivalently as if every symbol were the output of a memoryless channel W.

Typically, channels are classified in families characterized by some continuous variable, ϵ, selected to parameterise its behaviour. The variable ϵ is chosen such that increasing values of ϵ imply a degraded version of the channel [17]. For example, the family of binary symmetric channels (BSC) is parameterised by the error rate and the family of additive white Gaussian noise (AWGN) channels by the noise variance. A channel $W_{\epsilon'}$ is a degraded, or noisier, version of the channel W_ϵ if:

$$p_{W_{\epsilon'}}(y'|x) = p_Q(y'|y)p_{W_\epsilon}(y|x) \tag{1}$$

where Q is some auxiliary channel.

Let the information rate be the proportion of non redundant symbols sent through a channel. A code $\mathcal{C}(n,k)$, defined by a parity check matrix H, transforms an string of k symbols in a codeword c of n symbols with k independent symbols and $n-k$ redundant symbols, and in consequence achieves an information rate, R, of k/n.

This parameterisation allows to study two related concepts: the capacity of a channel, that is the maximum information rate that can be transmitted for a fixed ϵ and, for a specific error correcting code $\mathcal{C}$, the maximum value ϵ_{max}, i.e. the noisiest channel for which a sender can reliably transmit information with $\mathcal{C}$. The relationship between both answers gives an idea of the efficiency of the code, or in other words, how close is the coding rate of a code to the optimal value.

We can measure, analogously, the efficiency f of an information reconciliation protocol as the relation between the length of the messages exchanged to reconcile the strings, $|M|$, and the theoretical minimum message length. The problem of information reconciliation in secret key agreement is formally equivalent to the problem of source coding with side information [18], or how should X be encoded in order to allow a decoder with access to side information Y to recover X. Thus, the minimum message length is given by the conditional entropy $H(X|Y)$, since given the decoder access to side information Y no encoding of X shorter than $H(X|Y)$ allows for reliable decoding [18]. We can define the efficiency of an information reconciliation procedure as:

$$f = \frac{|M|}{H(X|Y)} \tag{2}$$

where $f=1$ stands, then, for perfect reconciliation.

Error correcting codes can be used for information reconciliation [19]. In information reconciliation, Y is already a noisy version of X (or viceversa) and the encoder and decoder have access to a noiseless channel. A code $\mathcal{C}$ can be used for information reconciliation using *syndrome* coding [20]. The syndrome of $\mathbf{x}$, an instance of X, is defined as $s(\mathbf{x}) = H \cdot \mathbf{x}$, with length per symbol $1-R$, indicates in which of the cosets of $\mathcal{C}$ is $\mathbf{x}$ a codeword. In other words, a decoder should recover $\mathbf{x}$ with high probability given access to $\mathbf{y}$, an instance of Y, and $s(\mathbf{x})$, the syndrome of $\mathbf{x}$, in a code adapted to the correlation between X and Y. Then, the efficiency of an information reconciliation method based in syndrome coding is given by:

$$f_{\mathcal{C}} = \frac{1-R}{H(X|Y)} \tag{3}$$

which in the special, but important, case of a binary symmetric channel with error rate ϵ, BSC(ϵ), can be written as:

$$f_{\mathrm{BSC}(\epsilon)} = \frac{1-R}{h(\epsilon)} \tag{4}$$

where $h(\epsilon)$ is the binary Shannon entropy.

The behaviour of the reconciliation efficiency using a code $\mathcal{C}$ as a function of the characteristic parameter, here the error rate ϵ, is shown in Fig. 1. The efficiency decreases in the range $\epsilon \in [0, \epsilon_{\mathrm{max}}]$. Since the redundancy is fixed for a ϵ range, it is excessive and far from optimal for good channels, i.e. low values of ϵ; as the light gray line shows for a reconciliation

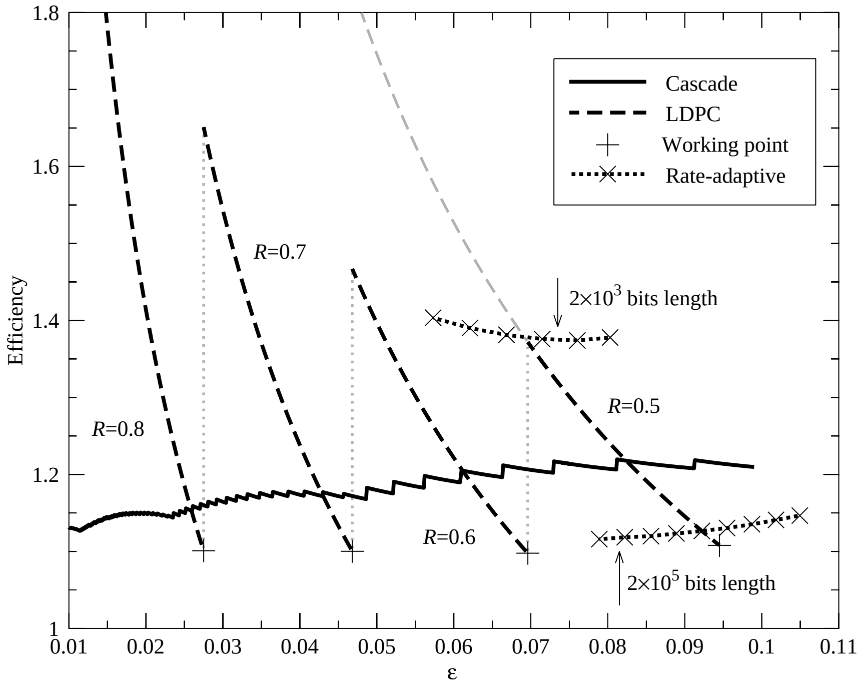

Fig. 1. Reconciliation efficiency using *Cascade* [4] and with LDPC codes for a 2×10^5 bits key length as a function of the channel error rate, ϵ. The short dashed curve marked as LDPC shows a set of codes where the best efficiency code is chosen for every value of ϵ. When ϵ is just below or at the "working" point, all the redundancy is used, producing high efficiency codes. As we move away from this point using the same code, the efficiency moves further from the optimal value. To put these data in context, we show with dotted lines the efficiency of a rate adaptive solution for R=0.5 using both, a short code of 2×10^3 bits length and a code of length 2×10^5. The proportion of symbols used to modulate the rate was set to 10%. Note that the length of the codes is two orders of magnitude smaller, which allows for hardware implementation. Since the rate modulation is performed by puncturing and shortening (simultaneously) the curve for the rate adaptive approach is centered on the curve for the non adaptive reconciliation with identical coding rate and length.

method based in a single LDPC code of 2×10^5 bits length. We can improve the reconciliation efficiency using a set of LDPC codes, as in the line marked LDPC, where we have chosen the code with the best efficiency for every value of ϵ, we observe a characteristic saw behaviour: the efficiency is good for ϵ values just below the $\epsilon_{\max}$ of every code and it degrades till the next code is used. This forces the use of many codes in order to cover a broad range of ϵ with good efficiency, not a very practical proposition. These two solutions based in LDPC codes are compared to a rate-compatible solution developed in the next section. The rate compatible solution has been calculated using short-length LDPC codes ($n = 2 \times 10^3$) which are suited for hardware implementations. There is a tradeoff between efficiency and code length, one of the aims of this work is to increase the reconciliation efficiency using short-length codes. In the figure, the solid line depicts the efficiency of *Cascade* [4], the *de facto* standard for information reconciliation in QKD. The *Cascade* protocol, though highly interactive, has the main advantage of being easy to implement and efficient enough for reconciling short strings. However, its interactivity can easily become a bottleneck in practical QKD systems, specially in those working at high speed or in a high QBER regime.

3 Rate-Adaptive Reconciliation

In the previous section we described a solution that allows to cover a ϵ range with several LDPC codes. However, even if a solution based in several LDPC codes is more efficient than a solution with just one code, it is still impractical. On one hand it forces Alice and Bob to store a set of codes, on the other it relies on precise estimations of ϵ: imprecisions in its estimation could lead to use a code unable to reconcile the strings. It would be highly desirable to use a single code able to adapt to different rates. In this section we review two techniques that allow to adapt the coding rate: *puncturing* and *shortening*.

3.1 Puncturing

Puncturing modulates the rate of a previously constructed code, $\mathcal{C}(n, k)$, by deleting a set of p symbols from the codewords, converting it into a $\mathcal{C}(n - p, k)$ code (see Refs. [21, 22]). The rate is then increased to:

$$R(p) = \frac{k}{n - p} = \frac{R_0}{1 - \pi} \tag{5}$$

where $R_0 = k/n$ is the rate of the original code and $\pi = p/n$ is the fraction of punctured symbols.

Syndrome coding can then be used to adapt the code rate in the following way. Let $\mathcal{C}(n, k)$ be a code that can correct noise up to $\epsilon_{\max}$ for some channel family and let X and Y be two m length strings, with $m = n - p$, correlated as if they were the input and output of a channel characterized by $\epsilon < \epsilon_{\max}$. The encoder can send the syndrome in $\mathcal{C}$ of a word $\widehat{X}$ constructed by embedding X in a string of size n and filling the other p positions with random bits. If the new coding rate, $R(p) = R_0/(1 - p)$ is adapted to ϵ the decoder should recover $\widehat{X}$ with high probability.

We can think of a reconciliation protocol based only in punctured codes: the parties would agree on an acceptable frame error rate (FER) and, depending on their estimation of the error rate, they would choose the value of p. If we consider the behaviour of FER as a function of ϵ for a set of fixed p values, as depicted in Fig. 2, this procedure can be regarded as moving along the horizontal axis from one code to the next. However, this way of proceeding has the shortcoming that if the channel is time varying (i.e. ϵ varies over time), the length of X and Y also varies to accommodate the different values of p needed to adapt the coding rate. We could think of scenarios where $m \gg n$ and X and Y can be divided in packets of length $n-p$ but this clearly does not apply to many situations.

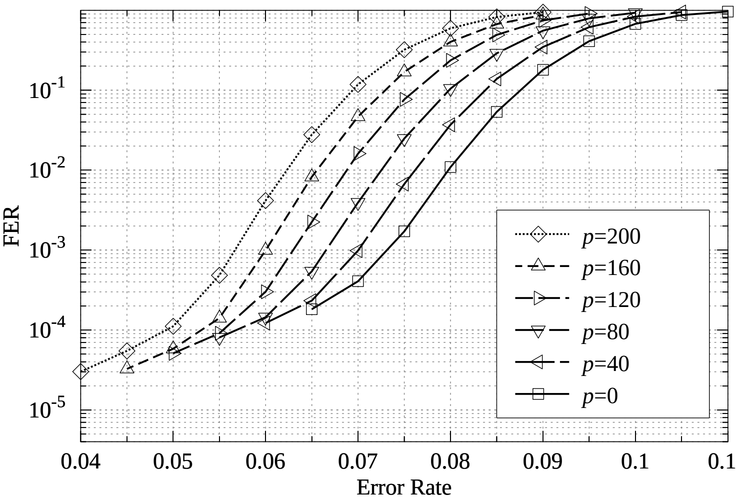

Fig. 2. Frame error rate over the BSC for a binary LDPC code of 2×10^3 bits length and rate $R_0 = 1/2$ as a function of the error rate. Several curves have been simulated for different proportions of punctured symbols. Due to the short code length, the distribution of punctured symbols has been intentionally chosen according to an optimized pattern [23]. Each point in the graph was calculated using as many codewords as needed till the FER was stable. Depending on the specific FER, this was typically between 10^3/FER and 10^2/FER.

3.2 Shortening

Puncturing increases the rate by reducing redundancy. The opposite is achieved through shortening: by increasing the redundancy, the information rate is reduced. This is done by fixing the value of a set of s symbols from the codewords in positions known to encoder and decoder. Shortening, then, converts a $\mathcal{C}(n,k)$ code in a $\mathcal{C}(n-s,k-s)$ one [24]. The result of simultaneously puncturing p symbols and shortening s symbols in the original code is thus a $\mathcal{C}(n-p-s,k-s)$ code with rate:

$$R = \frac{k-s}{n-p-s} = \frac{R_0 - \sigma}{1-\pi-\sigma} \qquad (6)$$

where $\sigma = s/n$ is the proportion of shortened symbols.

Typically only puncturing or shortening are used to adapt the rate of a code. However, when using syndrome coding over time varying channels, using just one of the two has the drawback that modifying the value of p or s implies modifying also the length of the reconciled strings with every code use. The combined application of both techniques allows to fix the size of the strings to reconcile and overcome this problem. In this case, a modulation parameter

$d = p + s$ can be set, thus fixing the lengths of X and Y to $n - d$ while allowing to modify p and s in order to adapt to different values of ϵ. Fig. 3 shows the performance of an error correcting code, again depicted as the FER versus the error rate of a BSC using both techniques simultaneously. A short-length LDPC code of 2×10^3 bits length was used.

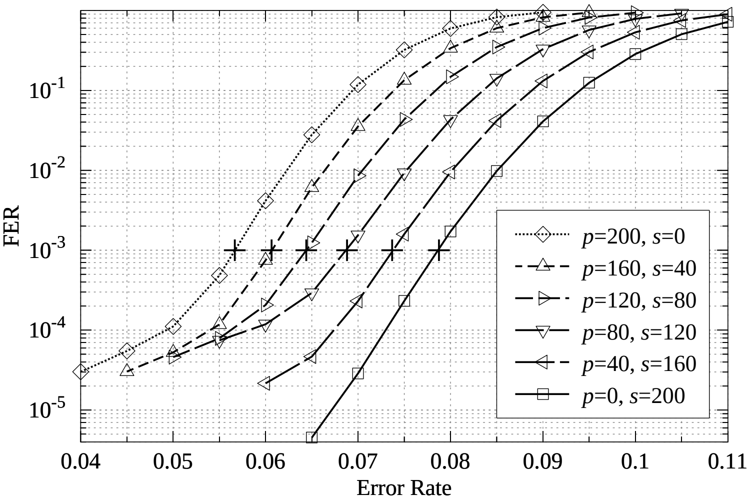

Fig. 3. FER over the BSC for a binary LDPC code of length 2×10^3 bits and rate $R_0 = 1/2$. Several curves have been simulated for different proportions of punctured and shortened symbols, with $d = 200$. The distribution of punctured symbols has been chosen according to a pattern previously estimated as proposed in Ref. [23].

If we call δ the proportion of punctured and shortened symbols, $\delta = d/n = \pi + \sigma$, a δ-*modulated* rate-adaptive code covers the range of rates $[R_{\min}, R_{\max}]$ defined by:

$$R_{\min} = \frac{R_0 - \delta}{1 - \delta} \leq R \leq \frac{R_0}{1 - \delta} = R_{\max} \tag{7}$$

There is a tradeoff between the covered error range, increasing with δ, and the efficiency of the procedure, decreasing with higher δ values [14].

The efficiency of a rate-adaptive protocol that systematically applies puncturing and shortening, one bit at a time, up to a pre-established δ is depicted in Fig. 1, marked as *Rate-adaptive*. Note how the saw tooth behaviour is eliminated. Note also that, since the codes used are very short ($n = 2 \times 10^3$), the efficiency is worse than that of *Cascade*.

4 Blind Reconciliation

In the rate-adaptive algorithm just outlined, the proportion, δ, of punctured plus shortened symbols is held constant. This proportion is calculated after an error rate (channel parameter) estimation. The only classical communication that is needed among Alice and Bob is one message from Alice to send the syndrome and the shortened information bits. This makes for a close to minimal interactivity protocol that is also highly efficient. Now, if we relax the interactivity condition and allow for a limited amount of communications, the panorama changes significantly.

Let us start by assuming again a value for δ covering the range of rates $[R_{\min}, R_{\max}]$ with the code with $R_{\min}$ able to correct words transmitted through the noisiest channel expected.

In a first message, Alice can include only the syndrome and no shortened bits, i.e. all the d symbols that can be either punctured or shortened, are punctured ($\pi = \delta$). If we look at Fig. 3, where we plot the behavior of FER as a function of ϵ using different proportions of punctured and shortened symbols, we can see that we are trying to correct errors with the code with the highest FER and highest rate, which is the one with $p = 200$.

If the reconciliation fails, no other information than the syndrome has been leaked, since punctured symbols do not disclose information. Alice can then reveal a set of the values of the previously punctured symbols. In this way the rate of the code is reduced, but the decoding success probability is increased. Returning to Fig. 3, this is like moving along the dotted vertical line and changing the code with $p = 200$, $s = 0$ ($\mathcal{C}(2000-200, 1000)$) by the code with $p = 160$, $s = 40$ ($\mathcal{C}(2000-200, 1000-40)$) and using it to correct the same string. Only the previously punctured but now shortened symbols reveal extra information. The protocol runs on the same string by revealing more information on the values of previously punctured symbols till success is achieved (or all the symbols were shortened without syndrome matching and it fails), effectively by using at each iteration codes with lower rate and FER.

In Fig. 4 we illustrate two iterations of the protocol in use to reconcile a string of length $m = 8$ using $d = 8$ extra symbols. It is also assumed that in every iteration $\Delta = 4$ symbols can be changed from punctured to shortened. In the first step, the m symbols are incremented with the $d = 8$ punctured ones to a total length of $n = m+d = 16$. At this point, the syndrome is calculated and the value sent to Bob. It is assumed that there is no syndrome match, hence the next iteration in which $\Delta = 4$ of the previously punctured symbols change to shortened. This information is sent to Bob. Again, a no match is assumed and the protocol proceeds to its second iteration, where another 4 symbols are revealed changing from punctured to shortened. Here the protocol ends, no matter whether there is a syndrome match or not, since all the punctured symbols have changed to shortened. If there is a syndrome match, then there is guarantee that the string $(x_1, x_2, \ldots, x_m)$ in the emitter side and $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m)$ in Bob's side are the same. Otherwise, the protocol fails for this string.

This whole procedure is done using the same base code and without needing an estimate for ϵ, hence the *blind* name. Only a rough estimate of the channel parameter is needed to design the base code. Note that this protocol requires some interactivity since, at each iteration in which there is no syndrome matching, a set of values for the shortened symbols must be communicated. As we show in the results section, a protocol with a very high average efficiency can be obtained using short codes and using only three iterations.

4.1 Blind protocol

We formally describe below the method for blind reconciliation outlined above. Note how there is no need of an a priori error estimate (except for the one implicitly embodied in the selection of the code $\mathcal{C}$) and a controlled amount of interactivity (t messages are exchanged at most).

Step 0: Set up.— Let $\mathcal{C}(n, k)$ be a code $\mathcal{C}$ that can correct noise up to $\epsilon_{\max}$ for some channel family. Let X and Y be two strings that two parties Alice and Bob wish to reconcile in at most t iterations. Let X and Y be of length m, with $m = n - d$, and every symbol of X

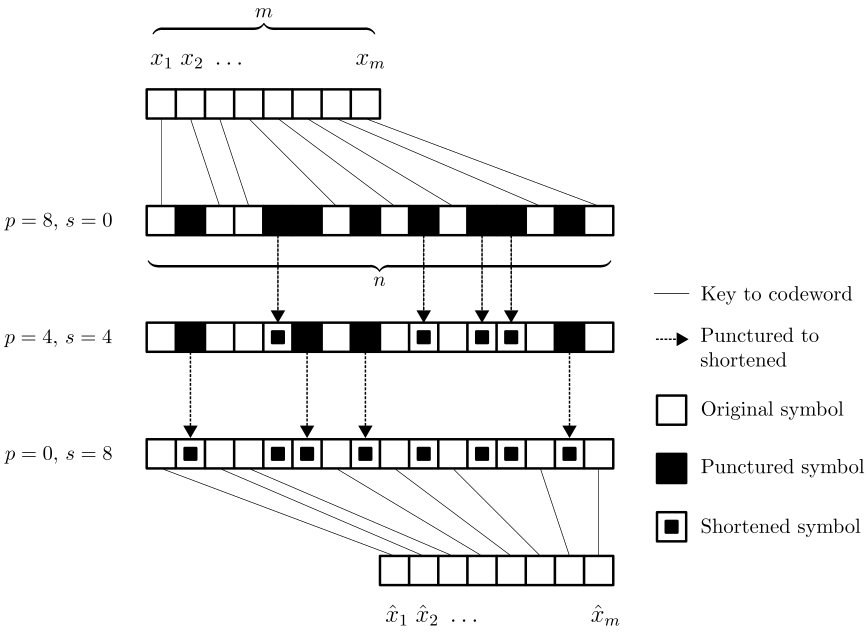

Fig. 4. Interactive protocol (blind), as described in the text.

and Y the input and output of a memoryless channel characterized by $\epsilon < \epsilon_{\max}$. Alice and Bob set $s = 0$, $p = d$ and $\Delta = d/t$. For simplicity in the description we assume $\Delta \in \mathbf{N}$.

Step 1: Encoding.— Alice sends the syndrome in $\mathcal{C}$ of a word $\widehat{X}$ consisting on embedding X in a length n string and filling the remaining d positions with random symbols.

Step 2: Decoding.— Bob constructs the word $\widehat{Y}$ consisting on the concatenation of Y the received s symbols and p random symbols. If Bob recovers X he reports success and the protocol ends.

Step 3: Re-transmission.— If $d = s$ the protocol fails, else Alice sets $s = s + \Delta$, reveals Bob Δ symbols and they return to Step 2 and perform a new iteration.

4.2 Average Efficiency

The average rate of the blind reconciliation protocol can be calculated as:

$$\hat{R} = \sum_{i=1}^{n} \alpha_i r_i \tag{8}$$

where α_i is the fraction, normalised to 1, of codewords that have been corrected in the iteration i. r_i is the information rate in the same iteration. Using Eq. (4) we obtain the expression for the average efficiency over the BSC(ϵ):

$$\hat{f}_{\mathrm{BSC}}(\epsilon) = \frac{1 - \hat{R}}{h(\epsilon)} = \frac{1 - \sum_{i=1}^{n} \alpha_i r_i}{h(\epsilon)} \tag{9}$$

Let $F^{(i)}$ be the FER when correcting with adapted rate r_i. Then the fraction of corrected codewords during the i-iteration is given by:

$$\alpha_i = \frac{F^{(i-1)} - F^{(i)}}{1 - F^{(n)}} \tag{10}$$

where $F^{(0)} = 1$.

Now, the average rate can be expressed as:

$$\hat{R} = \sum_{i=1}^{n} \frac{F^{(i-1)} - F^{(i)}}{1 - F^{(n)}} \cdot r_i = \frac{r_1 - F^{(n)} r_n}{1 - F^{(n)}} + \sum_{i=1}^{n-1} \frac{F^{(i)}}{1 - F^{(n)}} (r_{i+1} - r_i) \tag{11}$$

where the second equality holds for $n \geq 2$. Assuming that in every iteration we translate a constant proportion of punctured symbols to shortened symbols, the information rate used during the i-iteration is given by:

$$r_i = \frac{R_0 - \sigma_i}{1 - \delta} \tag{12}$$

where σ_i is the fraction of shortened symbols during the iteration i, such that $\sigma_1 = 0$ and $\sigma_n = \delta$. The rate increment between two consecutive iterations is also constant:

$$r_{i+1} - r_i = \frac{-\delta/(n-1)}{1 - \delta} \tag{13}$$

Let us define $\beta = \delta/(1-\delta)$ and then $r_{i+1} - r_i = -\beta/(n-1)$. The average rate can be now written as:

$$\hat{R} = \frac{r_1 - F^{(n)} r_n}{1 - F^{(n)}} - \frac{\beta}{n-1} \sum_{i=1}^{n-1} \frac{F^{(i)}}{1 - F^{(n)}} = r_1 + \frac{\beta}{1 - F^{(n)}} \left(F^{(n)} - \frac{1}{n-1} \sum_{i=1}^{n-1} F^{(i)} \right) \tag{14}$$

Where we have taken into account that in the first iteration every selected symbol is punctured (hence $r_1 = R_{max}$), while in the last one every selected symbol is shortened (hence $r_n = R_{min}$). The first and last coding rate, r_1 and r_n, are then given by:

$$R_{max} \equiv r_1 = \frac{R_0}{1 - \delta}; \quad R_{min} \equiv r_n = \frac{R_0 - \delta}{1 - \delta} = r_1 - \beta \tag{15}$$

Note that in the rate-adaptive approach a typical value for the frame error rate in a reliable reconciliation is 10^{-3}; i.e. we can then neglect the last contribution for the FER ($F^{(n)} \approx 0$), and thus the average rate is given by:

$$\hat{R} \approx r_1 - \frac{\beta}{n-1} \sum_{i=1}^{n-1} F^{(i)} \tag{16}$$

We describe a rough estimate of the frame error rate, $F(i)$, of linear codes in Appendix A. This analytical technique, known as the Gaussian approximation, captures the behavior of LDPC codes in the region where it transitions from correcting almost everything to correcting almost nothing (the waterfall region). Since the Gaussian approximation takes only into account the threshold and code length, it cannot be expected to reproduce the numerical values with the same accuracy than a numerical simulation with a big sample [36]. It is, however, much faster. Using this approach we approximate the behaviour of a finite-length LDPC code without having to perform heavy computer simulations.

In Fig. 5 we compare the decoding threshold (bullets), the rate-adaptive non-iterative protocol (boxes) and the average efficiency of the blind protocol for a different number of iterations calculated using Eqs. (9) and (16). The figure shows the estimated efficiency for three different short-length LDPC codes of 2×10^3 bits in the error rate range $\epsilon \in [0.02, 0.11]$ with the proportion of punctured and shortened symbols set to $\delta = 10\%$. From top to bottom, the coding rates are $R = 0,5$, $R = 0,6$ and $R = 0,7$.

Since the Gaussian approximation is a function of the decoding threshold, the maximum granularity is limited by the number of computed thresholds. We computed 21 decoding thresholds using the density evolution algorithm described in Ref. [17]. We stopped at 21 iterations because the curve had already converged.

The figure shows that when the error rate increases, the blind protocol adapts its behavior to the channel parameter, ϵ. However, the efficiency deteriorates for low error rates. More markedly for higher coding rates.

In the figure we can differentiate three regions. First, when every symbol is punctured the redundancy is fixed, in consequence the efficiency of both, the blind protocol and the rate adaptive, coincide up to the point marked A. Second, the main region covers the central area: from point A in the figure to the right. In this region, the blind protocol shows a better efficiency than the rate-adaptive protocol even for a low number of iterations. And third, the end of the correctable region, that depends on the maximum acceptable FER, as marked in the figure with a vertical line. In this point every symbol is shortened, i.e. this is the code with the lowest information rate.

Depending on the number of iterations allowed, which limits its maximum interactivity, the figure shows that the protocol can approach the decoding threshold. The best reconciliation efficiency for the blind protocol is achieved when it runs with $\Delta = 1$, i.e. in every iteration only one punctured symbol is converted to a shortened one. This intuitively holds because every extra intermediate code has a non zero probability of correcting which in turn increases the average coding rate. For instance, when using a code of length 2×10^3 and $\delta = 10\%$ this maximum number of iterations would be 200.

We cover in Fig. 6 the lower part of the error rates of interest in QKD. In this figure we compare the performance of two LDPC codes with coding rate $R = 0.8$ but different lengths (2×10^3 (top) and 10^4 (down) bits). The larger LDPC code was used to palliate the loss of efficiency. Codes of length 10^4 bits can still be implemented in hardware, although with a small throughput penalty [34]. Note that the value that δ takes has an impact in the achievable efficiency as shown in Ref. [14]. In consequence, we additionally reduce δ from 10% to 5% to improve the efficiency with high coding rates, thereby reducing the error rate range covered.

5 Simulation Results

Simulation results were computed to compare the protocol proposed in Ref. [14] with the blind protocol, but using short-length LDPC codes. These simulations were performed for several error rate ranges, covering both low and high values, over the binary symmetric

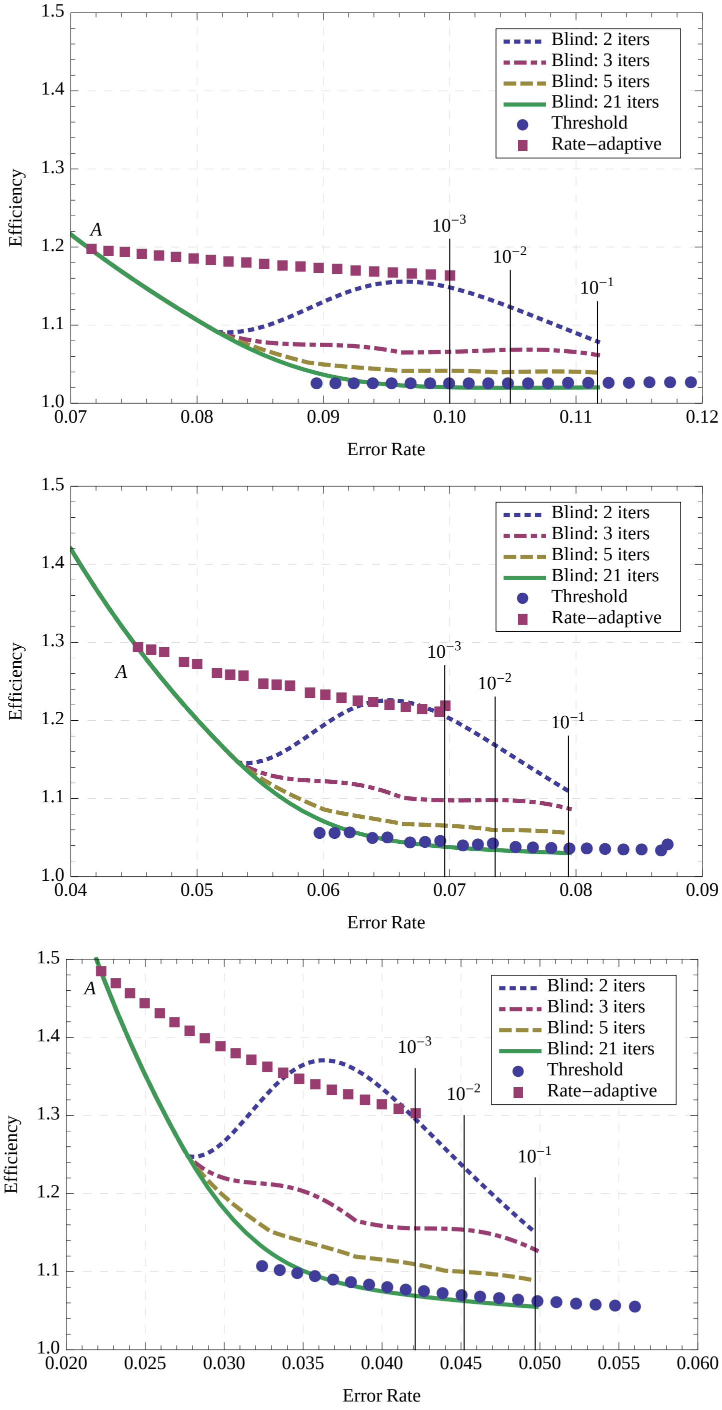

Fig. 5. The estimated efficiency (Gaussian approximation) of the rate adaptive protocol is compared to the efficiency of the blind protocol for short codes (2×10^3 bits length). Three coding rates are used to cover the error range $[2.5\%, 11\%]$. A detailed description is given in the text.

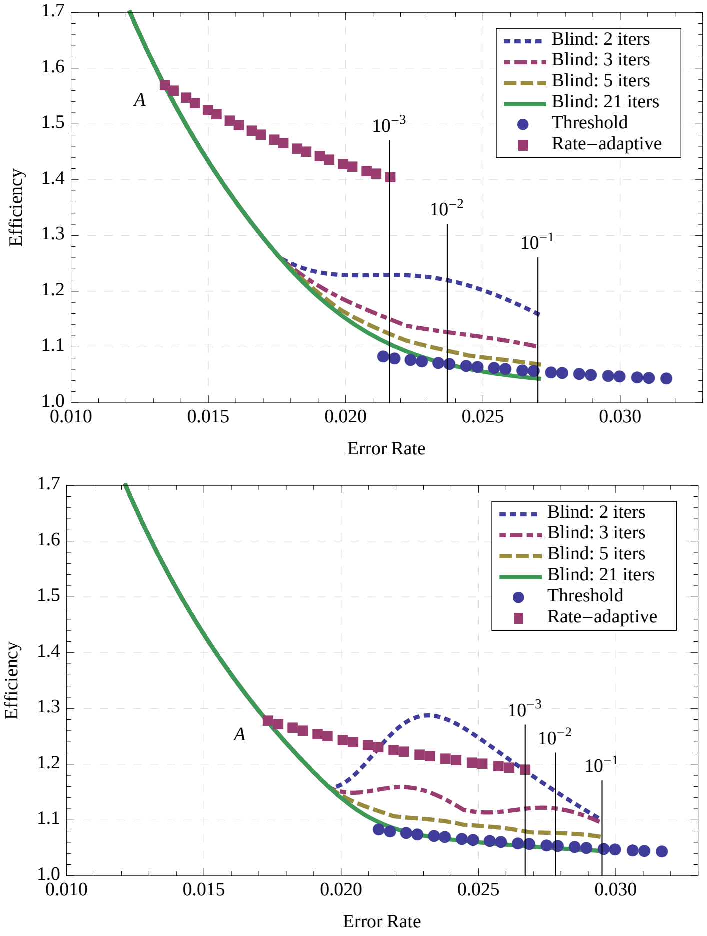

Fig. 6. Estimated efficiency curves, as in Fig. 5, for a coding rate $R = 0,8$ and different codeword length, 2×10^3 (top) and 10^4 (down) bits length. A detailed description is given in text.

channel (BSC). Each simulated point was iterated till the FER was stable within a fraction of its average value. This termination condition was usually achieved when decoding between 10^2/FER and 10^3/FER. Hence each point is the result of a very time consuming procedure that decodes between 10^5 and 10^8 codewords, depending on the FER. An LDPC decoder based on the sum-product algorithm was used. New LDPC codes were designed for several coding rates (see Appendix B) and constructed using Ref. [27]. This construction focus in reducing the error floor.[b] The objective of considering low error floor construction was to improve the efficiency of the blind protocol as it corrects words in a wide error rate region.

The shortened symbols were selected randomly while the punctured symbols were selected according to a computed pattern for intentional puncturing as described in Ref. [26]. This intentional puncturing algorithm is specifically designed for moderate puncturing rates, i.e. low values of δ.

Fig. 7 shows the efficiency, as defined in Eq. (2), of the rate-adaptive protocol proposed in Ref. [14], and the average efficiency of the blind protocol for two different rates. The blind protocol is simulated with two different maximum number of iterations or equivalently with two different values of Δ. The first one, a lightweight version limited to a maximum of 3 iterations, is compared with the maximally interactive version where in every iteration only one punctured symbol becomes a shortened one ($\Delta = 1$). Simulations were computed using an LDPC code of 2×10^3 bits length with $\delta = 10\%$ and two coding rates $R = 0.5$ and $R = 0.6$.

The simulations for the rate-adaptive approach were computed with an acceptable FER set to 10^{-3}. However, in this figure and in the following one, the curves for the blind protocol extend beyond the acceptable FER. To show its value, in the version with a maximum of three iterations the average FER is printed.

The figure shows how the efficiency improves with interactivity (more iterations) and with the error rate. The efficiency in Fig. 7 also coincides for the rate-adaptive and interactive approaches for error rates below A, a behaviour similar to Fig. 5.

In Fig. 8 the efficiency is studied in the low error rate range in QKD. An LDPC code of 10^4 bits length and coding rate $R = 0.8$ is used. Due to this high coding rate, only 5% of the symbols were selected for puncturing and shortening. The figure shows that the average efficiency quickly improves with the blind protocol, even when using only three iterations.

If we try to increase the range of error rates covered, we can increase the proportion of punctured and shortened symbols (see Eq. (7)). The results are shown in the bottom panel of Fig. 8, where the proportion is set to the same 8%, the maximum achievable value following the intentional puncturing proposal described in Ref. [26]. We observe that for a fixed number of iterations the efficiency is worse, as is clearly seen when comparing the dotted line (with a maximum of three iterations) in both panels. As expected, the efficiency for the maximum number of iterations, d, as it grows, improves.

The increase in efficiency with the number of iterations opens the possibility of having both, high efficiency and high throughput. The new generation of QKD systems are approaching sifted-key rates close to 1 Mbps [28–33]. Implementing real time error correction to provide secret keys at this speed is a challenging problem where a high throughput procedure with minimal communications is needed. Using *Cascade* under these constraints is unfeasible unless an extremely low latency network is used. Short-length LDPC codes have been implemented in

[b] The residual error in the low error rate region.

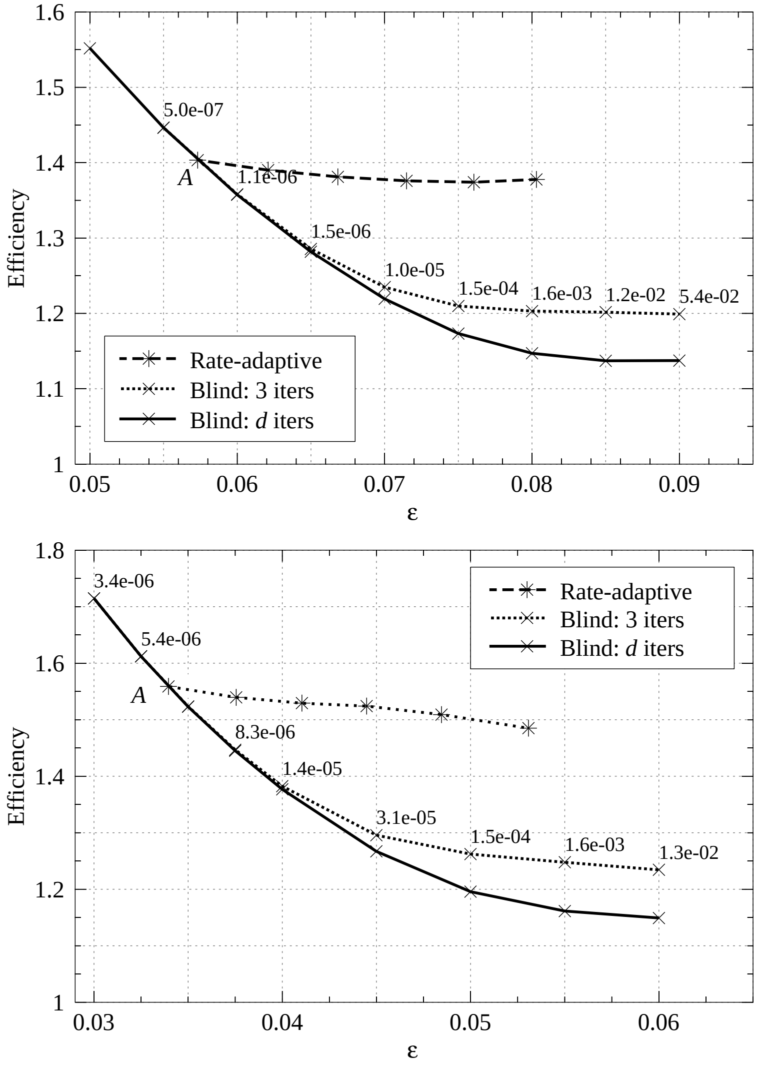

Fig. 7. Simulated curves for the rate-adaptive protocol proposed in Ref. [14] and the blind protocol. LDPC codes of 2×10^3 bits length and rates $R = 0.5$ (top) and $R = 0.6$ (bottom) were used. Punctured symbols were selected according to a pattern for intentional puncturing as proposed in Ref. [26]. The average FER is printed for each point of the three iterations curve. The lowest curve is for the $d = n * \delta = 200$ iterations. Every point for the rate-adaptive protocol was determined for a frame error rate of 10^{-3}.

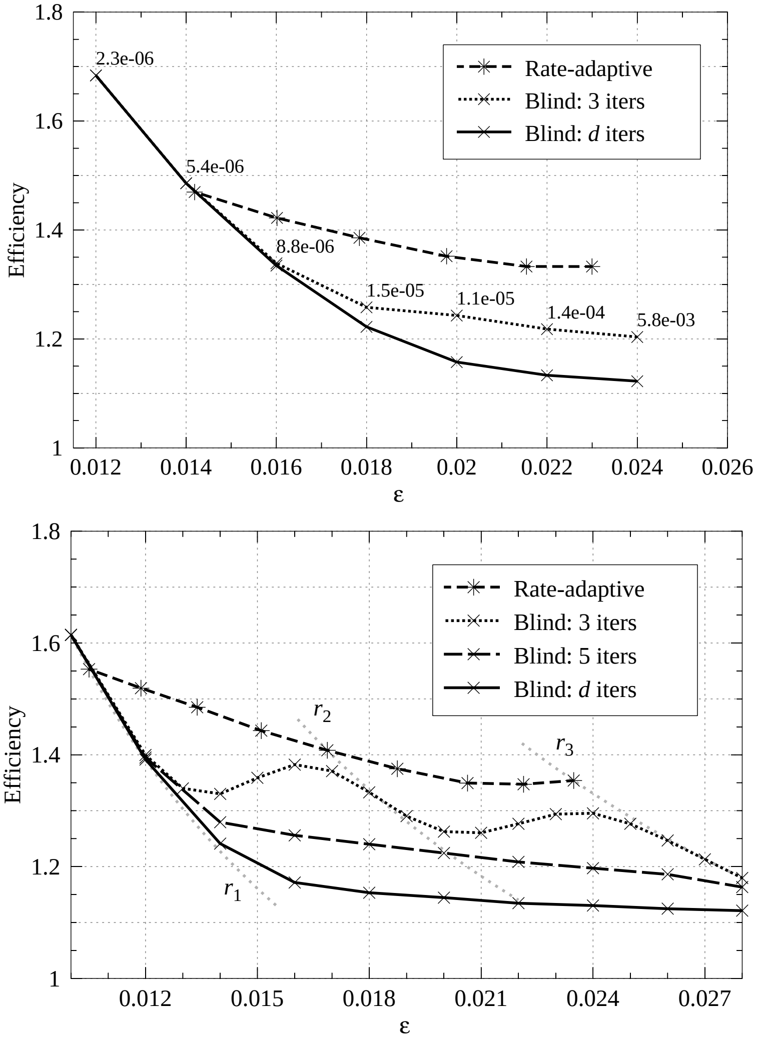

Fig. 8. Simulated curves for the rate-adaptive protocol proposed in Ref. [14] and the blind protocol. An LDPC code of 10^4 bits length and rate 0.8 was used. Symbols were selected according to an improved pattern for intentional puncturing as proposed in Ref. [26]. The average FER is printed for each point of the three iterations curve (top). The lowest curve is for $d = n * \delta = 1000$ iterations. Every point for the rate-adaptive protocol was determined for a frame error rate of 10^{-3}. In order to understand the behaviour of the curve for the blind protocol with a maximum of 3 iterations, we also plot (bottom) the efficiency of using (fixed-rate) LDPC codes with the coding rates associated with each iteration: $r_1 = \frac{R_0}{1-\delta}$, $r_2 = \frac{R_0-\delta/2}{1-\delta}$ and $r_3 = \frac{R_0-\delta}{1-\delta}$.

hardware for other purposes, like wireless networks [35], where they have demonstrated to be an excellent solution. In the QKD case, to use LDPC codes required to have codes designed for different error rates, thus making the process more complex and memory constrained. With the protocol presented here, the error estimation phase is not needed. The procedure can start directly and, if it fails, allowing a few iterations increases considerably the success probability. The price to pay is an extra message per failure. As shown in Figs. 5 and 8, the process converges quickly and only a few iterations are needed to increase the reconciliation efficiency significantly. This also avoids the need to store many precalculated codes.

In a hardware implementation, the iterations are easily realised just by copying the same functional decoder block as many times as the number of desired iterations. The string to be reconciled would start the iteration i in the first hardware block. If the decoding fails, the next hardware block would continue processing in a pipeline fashion, since computation and communication can be arranged in a way such that the disclosed symbols would arrive packed in the same message than the syndrome of the following strings to reconcile. The new syndrome would start being processed in the first hardware block while the second would continue working on the second iteration on the previous string. This pipeline can increase the efficiency while maintaining a high and —mostly— constant throughput at the expense of some extra hardware.

6 Conclusions

Results show that efficient reconciliation can be achieved using short-length LDPC codes and a slightly interactive protocol. Codes as short as 2000 bits are suitable for blind reconciliation. Even a minimum interactive protocol with a maximum of three iterations improves considerably the efficiency in high and low error rate regimes.

This protocol can be easily implemented in hardware and it can be also pipelined to increase the throughput of reconciled key. Its requirements are low enough to allow an implementation using an embedded processor within a compact, industrial grade QKD device.

The protocol presented here could improve the final secret-key length in those scenarios where part of the raw key has to be disclosed in order to estimate the channel parameter. This improvement lies in the fact that the protocol works without an error rate estimate.

Acknowledgements

The authors wish to thank Andreas Burg for helpful discussions about the hardware implementation of LDPC codes. The authors also gratefully acknowledge the computer resources, technical expertise and assistance provided by the *Centro de Supercomputación y Visualización de Madrid*[c](CeSViMa) and the Spanish Supercomputing Network.

This work has been partially supported by the project Quantum Information Technologies in Madrid[d](QUITEMAD), Project P2009/ESP-1594, *Comunidad Autónoma de Madrid*.

References

1. N. Gisin, G. Ribordy, W. Tittel and H. Zbinden (2002), *Quantum cryptography*, Rev. Mod. Phys., Vol. 74, pp. 145-195.

[c] http://www.cesvima.upm.es
[d] http://www.quitemad.org

2. D. Gottesman, H.-K. Lo, N. Lutkenhaus and J. Preskill (2004), *Security of quantum key distribution with imperfect devices*, Quantum Inform. Comput., Vol. 4, No. 5, pp. 325-360.
3. C.H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin (1992), *Experimental quantum cryptography*, J. Cryptology, Vol. 5, No. 1, pp. 3-28.
4. G. Brassard and L. Salvail (1994), *Secret-Key Reconciliation by Public Discussion*, in Eurocrypt'93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology, Lecture Notes in Computer Science, Vol. 765, pp. 410-423.
5. C.H. Bennett, G. Brassard, C. Crepeau and U.M. Maurer (1995), *Generalized privacy amplification*, IEEE Trans. Inf. Theory, Vol. 41, No. 6, pp. 1915-1923.
6. C.H. Bennett, G. Brassard and J.M. Roberts (1988), *Privacy Amplification by Public Discussion*, SIAM J. Comput., Vol. 17, No. 2, pp. 210-229.
7. M. Van Dijk and A. Koppelaar (1997), *High Rate Reconciliation*, in ISIT 1997, IEEE International Symposium on Information Theory, p. 92.
8. T. Sugimoto and K. Yamazaki (2000), *A study on secret key reconciliation protocol "Cascade"*, IEICE Trans. Fundam. Electron. Commun. Comput. Sci., Vol. E83-A, No. 10, pp. 1987-1991.
9. S. Liu, H.C.A. Van Tilborg and M. Van Dijk (2003), *A Practical Protocol for Advantage Distillation and Information Reconciliation*, Designs Codes Cryptogr., Vol. 30, No. 1, pp. 39-62.
10. W.T. Buttler, S.K. Lamoreaux, J.R. Torgerson, G.H. Nickel and C.G. Peterson (2003), *Fast, efficient error reconciliation for quantum cryptography* Phys. Rev. A, Vol. 67, No. 5, p. 052303.
11. C. Elliott, D. Pearson and G. Troxel (2003), *Quantum Cryptography in Practice*, in Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 227-238.
12. D. Pearson (2004), *High-speed QKD Reconciliation using Forward Error Correction*, in 7th International Conference on Quantum Communication, Measurement and Computing, Vol. 734, pp. 299-302.
13. C. Elliott, A. Colvin, D. Pearson, O. Pikalo, J. Schlafer and H. Yeh (2005), *Current status of the DARPA Quantum Network*, quant-ph/0503058.
14. D. Elkouss, J. Martinez-Mateo and V. Martin (2011), *Information Reconciliation for Quantum Key Distribution*, Quantum Inform. Comput., Vol. 11, No. 3&4, pp. 226-238.
15. J. Martinez-Mateo, D. Elkouss and V. Martin (2010), *Interactive Reconciliation with Low-Density Parity-Check Codes*, 6th Int. Symposium on Turbo Codes & Iterative Information Processing (ISTC'2010), pp. 270-274.
16. G. Van Assche (2006), *Quantum Cryptography and Secret-Key Distillation*, Cambridge University Press.
17. T.J. Richardson and R.L. Urbanke (2001), *The capacity of low-density parity-check codes under message-passing decoding*, IEEE Trans. Inf. Theory, Vol. 47, No. 2, pp. 599-618.
18. D. Slepian and J. Wolf (1973), *Noiseless coding of correlated information sources*, IEEE Trans. Inf. Theory, Vol. 19, No. 4, pp. 471-480.
19. C. Crépeau (1995), *Réconcilliation et Distillation publiques de secret*, unpublished manuscript, available at http://www.cs.mcgill.ca/˜crepeau/theses.html.
20. T.M. Cover and J.A. Thomas (1991), *Elements of information theory*, Wiley-Interscience.
21. J. Ha, J. Kim and S.W. McLaughlin (2004), *Rate-compatible puncturing of low-density parity-check codes*, IEEE Trans. Inf. Theory, Vol. 50, No. 11, pp. 2824-2836.
22. H. Pishro-Nik and F. Fekri (2007), *Results on Punctured Low-Density Parity-Check Codes and Improved Iterative Decoding Techniques*, IEEE Trans. Inf. Theory, Vol. 53, No. 2, pp. 599-614.
23. J. Ha, J. Kim, D. Klinc and S.W. McLaughlin (2006), *Rate-compatible punctured low-density parity-check codes with short block lengths*, IEEE Trans. Inf. Theory, Vol. 52, No. 2, pp. 728-738.
24. T. Tian and C.R. Jones (2005), *Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing*, EURASIP J. Wirel. Commun. Netw., Vol. 2005, No. 5, pp. 789-795.
25. X.-Y. Hu, E. Eleftheriou and D.M. Arnold (2005), *Regular and irregular progressive edge-growth tanner graphs*, IEEE Trans. Inf. Theory, Vol. 51, pp. 386-398.

26. D. Elkouss, J. Martinez-Mateo and V. Martin (2010), *Untainted Puncturing for Irregular Low-Density Parity-Check Codes*, submitted to IEEE Commun. Lett., arXiv:1103.6149 [cs.IT].
27. Sung-Ha Kim, Joon-Sung Kim and Dae-Son Kim (2007), *LDPC Code Construction with Low Error Floor Based on the IPEG Algorithm*, Vol. 11, No. 7, pp. 607-609.
28. H. Takesue, E. Diamanti, T. Honjo, C. Langrock, M.M. Fejer, K. Inoue and Y. Yamamoto (2005), *Differential phase shift quantum key distribution experiment over 105 km fibre*, New J. Phys., Vol. 7, No. 1, p. 232.
29. X. Tang, L. Ma, A. Mink, A. Nakassis, H. Xu, B. Hershman, J.C. Bienfang, D. Su, R.F. Boisvert, C.W. Clark and C.J. Williams (2006), *Experimental study of high speed polarization-coding quantum key distribution with sifted-key rates over Mbit/s*, Opt. Express, Vol. 14, No. 6, pp. 2062-2070.
30. Z.L. Yuan, A.R. Dixon, J.F. Dynes, A.W. Sharpe and A.J. Shields (2008), *Practical gigahertz quantum key distribution based on avalanche photodiodes*, New J. Phys., Vol. 11, No. 4, p. 045019.
31. I. Choi, R.J. Young and P.D. Townsend (2010), *Quantum key distribution on a 10Gb/s WDM-PON*, Opt. Express, Vol.18, No. 9, pp. 9600-9612.
32. J. Zhang, P. Eraerds, N. Walenta, C. Barreiro, R. Thew, Rob and H. Zbinden (2010), *2.23 GHz gating InGaAs/InP single-photon avalanche diode for quantum key distribution*, in Proc. SPIE, Vol. 7681, 76810Z.
33. G.S. Buller, R.J. Collins, P.J. Clarke, P.D. Townsend (2010), *Gigahertz quantum cryptography*, in Proc. Communications and Photonics Conference and Exhibition (ACP), pp. 659-660.
34. C. Roth, A. Cevrero, C. Studer, Y. Leblebici and A. Burg (2011), *Area, throughput, and energy-efficiency trade-offs in the VLSI implementation of LDPC decoders*, in IEEE International Symposium on Circuits and Systems, pp. 1772-1775.
35. 802.11 Working Group of the 802 Committee (2009), *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Std 802.11n-2009.
36. R. Yazdani and M. Ardakani (2009), *Waterfall Performance Analysis of Finite-Length LDPC Codes on Symmetric Channels*, IEEE Trans. Commun., Vol. 57, No. 11, pp. 3183-3187.

Appendix A Frame Error Rate Analysis

In this work we study the FER for finite-length communications using the concept of *observed* channel introduced in Ref. [36]. Though a simple approach, it provides with an acceptable accuracy. The method is based on the analysis of the probability density function (pdf) of random variables corresponding to N received symbols. This pdf for a N finite-length code is compared with its estimated threshold, ε^*, such that an average frame error rate probability is calculated as the probability that the "observed" channel behaves worse than the code's decoding threshold. This observed channel is interpreted as the measurement of N samples, where each sample (bit) is an error with probability p.

Observed Channel.— Any communication channel (discrete memoryless channel) is stochastically modelled by a set of parameters. For instance, the binary symmetric channel (BSC) is parameterised by its error rate ϵ. However, these parameters accurately describe the behaviour of the modelled channel only in the asymptotic case, i.e. assuming infinite length communications. In the BSC(ϵ) we define the *observed* bit error rate in a communication, P_{obs}, as the number of errors divided by the length of this communication, N. This observed value is constant only in the asymptotic case, i.e. $P_{\text{obs}} = \epsilon \; \forall N$ only when $N \to \infty$. The distribution of errors in our *observed* BSC channel is then described by the following probability mass function (pmf):

$$f_{P_{\text{obs}}}(\epsilon, N, x) = \binom{N}{Nx} \epsilon^{Nx} (1-\epsilon)^{N-Nx} \tag{A.1}$$

where Nx is the number of errors in the communication of length N.

Assuming that the length of the communication is large enough (i.e. when it is higher than a few thousand bits) this pmf can be approximated with high precision by using a (continuous) Gaussian probability density function centered around the error rate ϵ and with variance $\sigma^2_{P_{\text{obs}}} = \epsilon(1-\epsilon)/N$:

$$f_{P_{\text{obs}}}(\epsilon, N, x) \approx \mathcal{N}(\mu_{P_{\text{obs}}}, \sigma^2_{P_{\text{obs}}}) \tag{A.2}$$

Frame Error Rate.— Let us now consider that we are using a finite-length linear code to correct any error occurred during the communication, then we can estimate the ratio of codewords that cannot be corrected by calculating the probability that the observed channel behaves worse than the decoding threshold of our code, ϵ^* (see Ref. [17]).

Using an error correction code of length N with a theoretical threshold of ϵ^*, the FER for our BSC(ϵ) channel can be reasonably approximated by:

$$\begin{aligned} F_{P_{\text{obs}}}(\epsilon, N, \epsilon^*) &= 1 - \Pr(P_{\text{obs}} \leq \epsilon^*) && \text{(A.3)} \\ &= \Pr(P_{\text{obs}} > \epsilon^*) = \int_{\epsilon^*}^{1} f_{P_{\text{obs}}}(\epsilon, N, x) dx && \text{(A.4)} \end{aligned}$$

Using the Gaussian approximation:

$$F_{P_{\text{obs}}}(\epsilon, N, \epsilon^*) \approx \frac{1}{\sqrt{2\pi\epsilon(1-\epsilon)/N}} \int_{\epsilon^*}^{1} e^{-\frac{N(x-\epsilon)^2}{2\epsilon(1-\epsilon)}} dx \tag{A.5}$$

Note that, for convenience, we have used the term F instead of $F_{P_{\text{obs}}}(\mathcal{C}(\theta), N, \epsilon^*)$ in the main body of the paper.

Note also that this analytical approximation is only valid for the behaviour in the *waterfall* region of an error correction code, since it does not include information about the performance in the error floor regime.

Appendix B Families of LDPC Codes

In this appendix we present the generating polynomials (see Ref. [17]) for the LDPC codes used in this paper. The design criteria was to maximise the threshold for a ratio of edges/bit not greater than 6.06. A small ratio of edges/bit reduces the achievable threshold but renders the codes more suitable for hardware implementations.

Coding rate $R = 0.5$, theoretical threshold $\epsilon^* = 0.102592$ (see Ref. [17]):

$$\begin{aligned} \lambda(x) &= 0.159673x + 0.121875x^2 + 0.11261x^3 + 0.190871x^4 + \\ &\quad 0.0770616x^9 + 0.337909x^{24} \\ \rho(x) &= 0.360479x^8 + 0.639521x^9 && \text{(B.1)} \end{aligned}$$

Coding rate $R = 0.6$, theoretical threshold $\epsilon^* = 0.0745261$:

$$\begin{aligned}\lambda(x) &= 0.11653x + 0.125646x^2 + 0.108507x^3 + 0.0534223x^4 + \\ &\quad 0.0727228x^6 + 0.0347964x^7 + 0.0729986x^8 + \\ &\quad 0.0752607x^{17} + 0.117103x^{31} + 0.223013x^{44} \\ \rho(x) &= 0.582731x^{13} + 0.417269x^{14}\end{aligned} \tag{B.2}$$

Coding rate $R = 0.7$, theoretical threshold $\epsilon^* = 0.0501875$:

$$\begin{aligned}\lambda(x) &= 0.091699x + 0.171401x^2 + 0.0683878x^3 + 0.120523x^4 + \\ &\quad 0.187471x^{10} + 0.208278x^{27} + 0.152239x^{29} \\ \rho(x) &= 0.806453x^{18} + 0.193547x^{19}\end{aligned} \tag{B.3}$$

Coding rate $R = 0.8$, theoretical threshold $\epsilon^* = 0.0289413$:

$$\begin{aligned}\lambda(x) &= 0.0667948x + 0.194832x^2 + 0.0570523x^3 + 0.0645024x^4 + \\ &\quad 0.204606x^8 + 0.0964409x^{14} + 0.23872x^{28} + 0.0770523x^{34} \\ \rho(x) &= 0.708874x^{29} + 0.291126x^{30}\end{aligned} \tag{B.4}$$